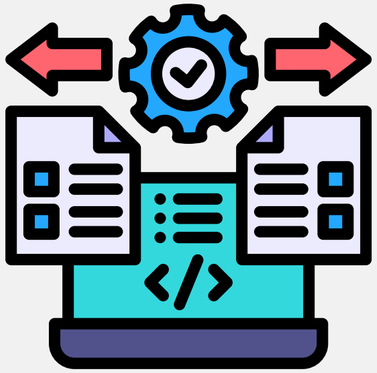
Tech Titans

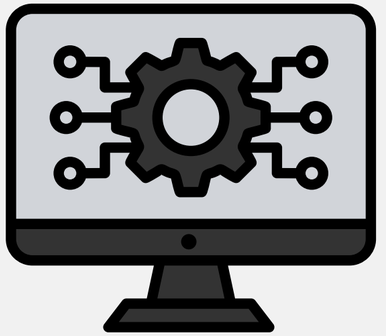
GROUP PROJECT

Git Like Version Control System

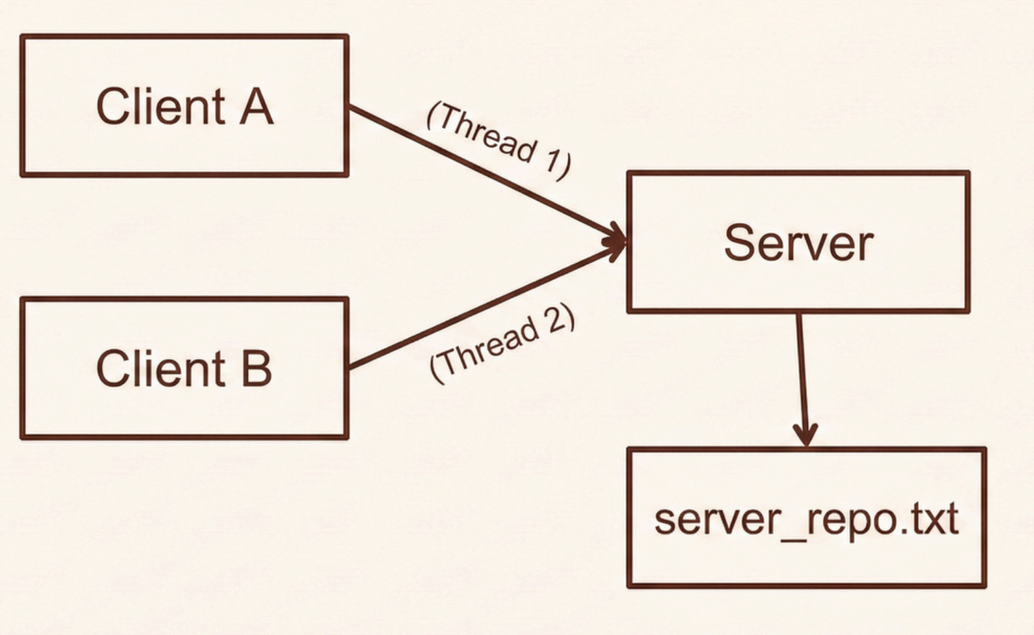


PROBLEM STATEMENT

- Design a simple Git like Version Control System where each change is pushed to a stack.
- Include Undo/Redo operation and branching using addition stack.



What is a Version Control System ???



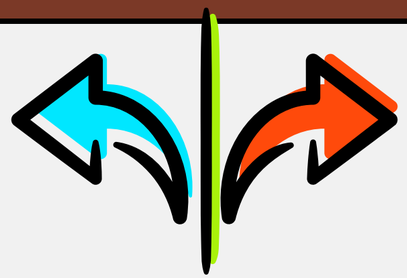
High-Level Architecture

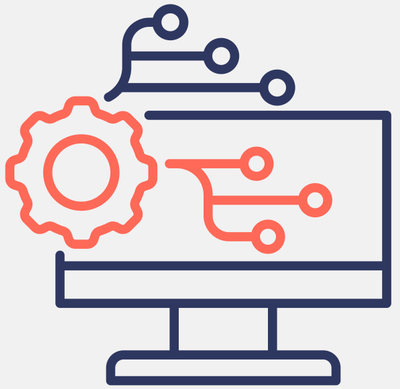
- **Client-Server Model**
- **Communication Protocol:** TCP/IP Sockets for reliable data transmission
- **Concurrency:** The server uses Multi-threading, allowing multiple clients to connect and work simultaneously without blocking each other.

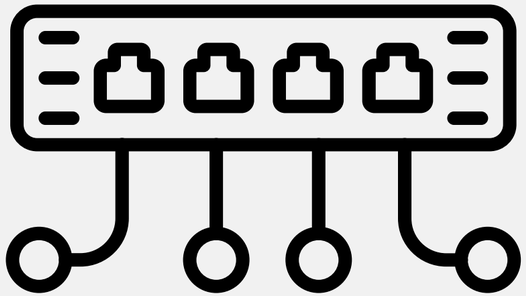
Feactures

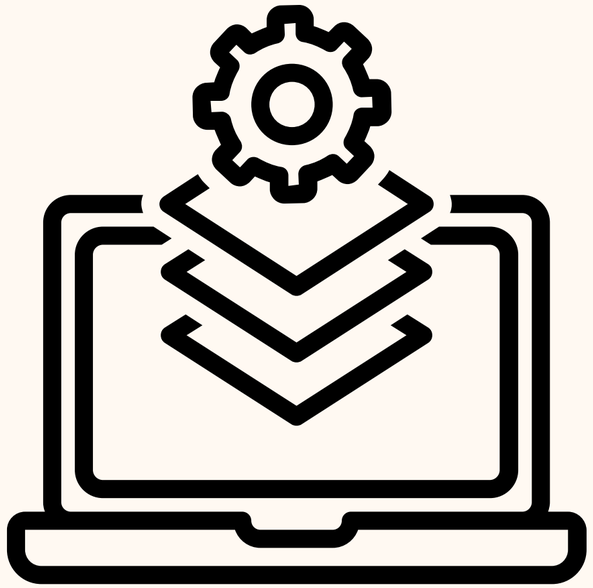
EDIT

- UNDO
- REDO
- COMMIT
- SHOW
- BRANCH : Creates a new branch
- CHECKOUT : Switch to a branch
- MERGE : merging of branch





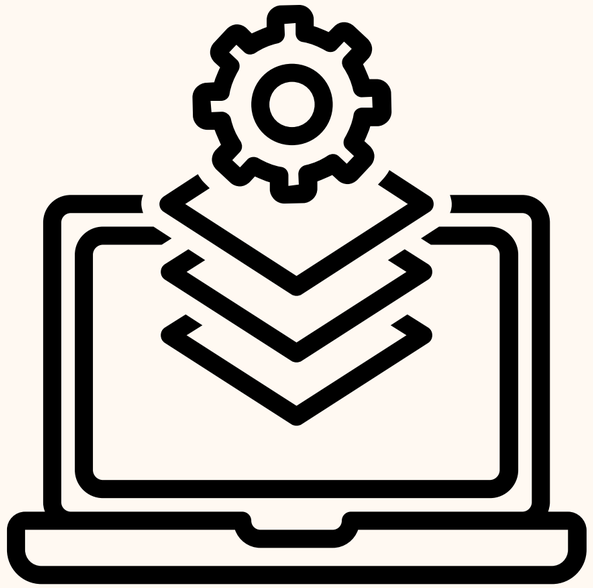




STACK

- A Linear Data Structure which follows LIFO Principle
- It is the perfect structure for linear history tracking

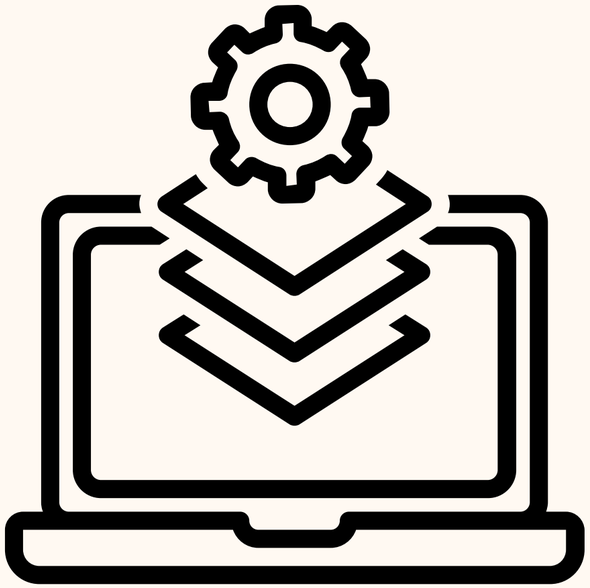
STACK IN VCS SYSTEM



Implementation:

- **push():** Add a new state.
- **pop():** Remove the most recent state (used for Undo).
- **peek():** View current state without removing it.

STACK IN VCS SYSTEM



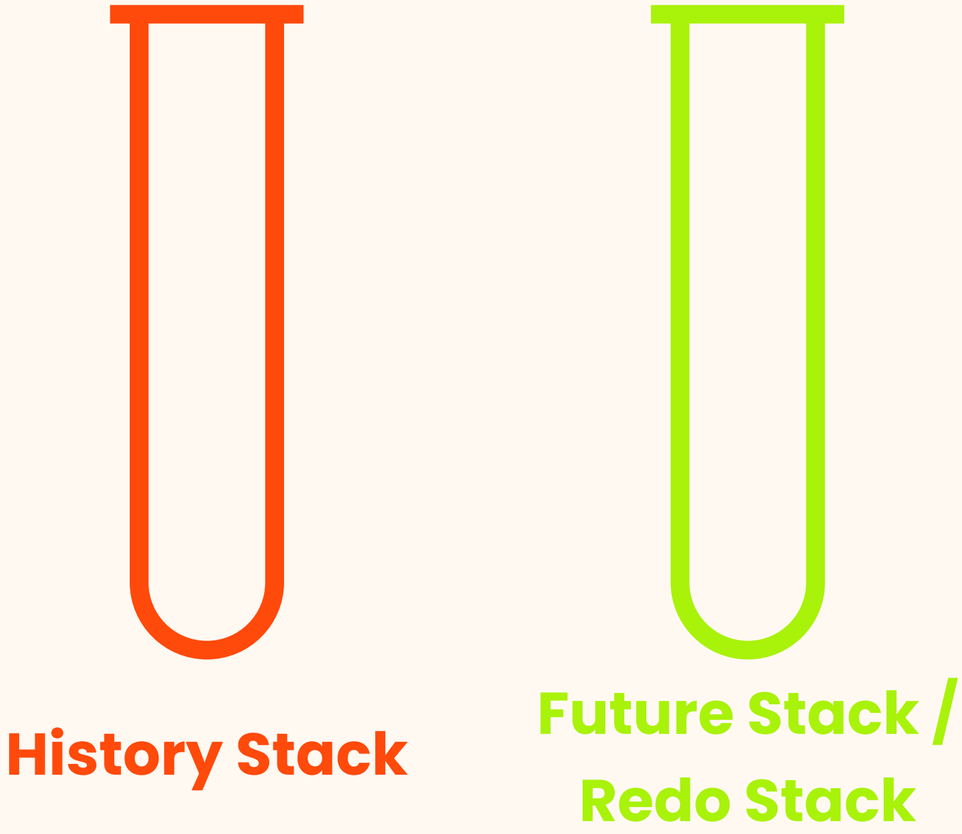
Error Handling:

- Custom **StackUnderFlowError** prevents errors when undoing an empty history.

STACKS USED IN VCS

Two-Stack Approach:

- **History Stack:** Stores every EDIT made.
- **Future Stack:** Stores states that were POPPED during an Undo.



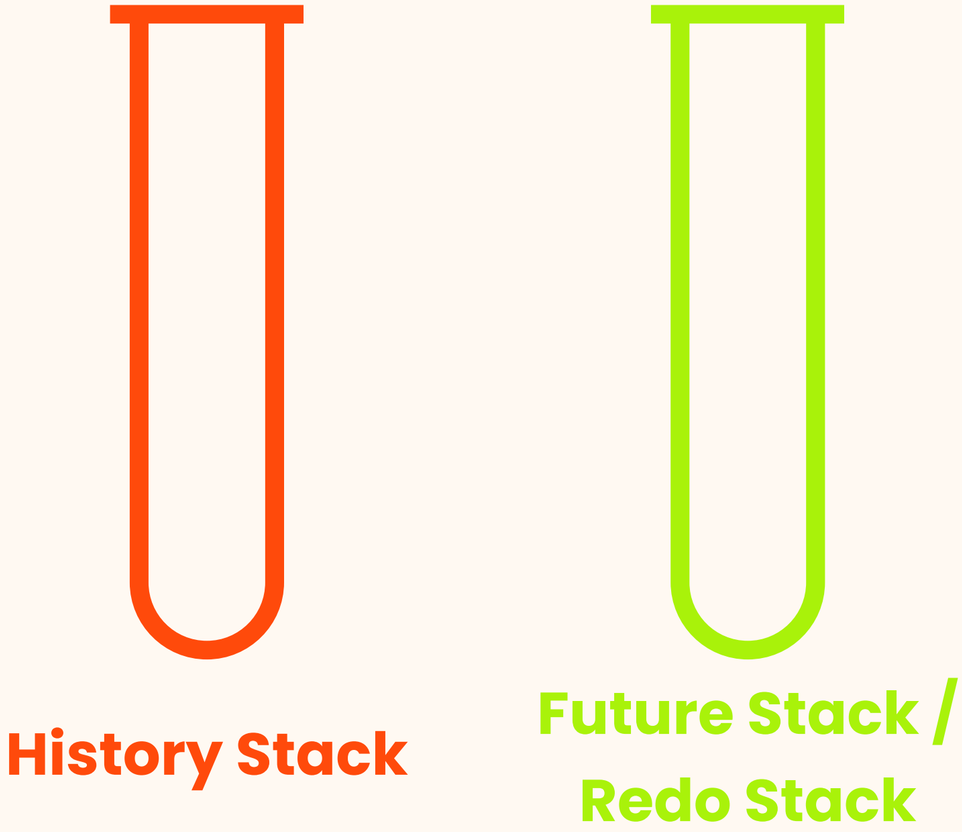
History Stack

Future Stack / Redo Stack

STACKS USED IN VCS

The Workflow:

- **On Edit:** Push new content to History Stack; Clear Future Stack.
- **On Undo:** Pop from History ➡️Push to Future.
- **On Redo:** Pop from Future ➡️ Push back to History.



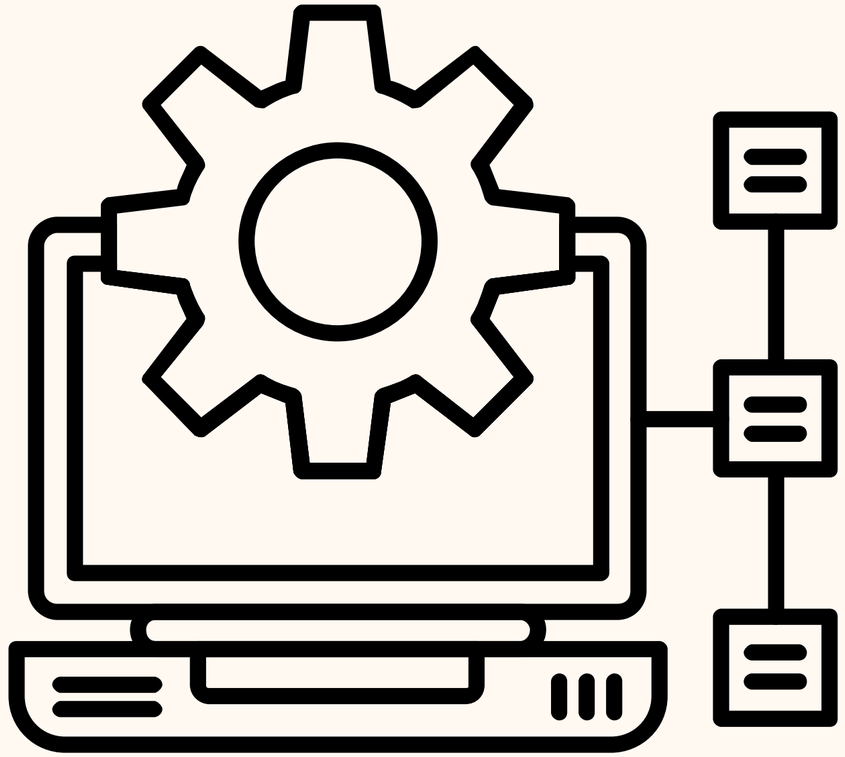
History Stack

Future Stack / Redo Stack

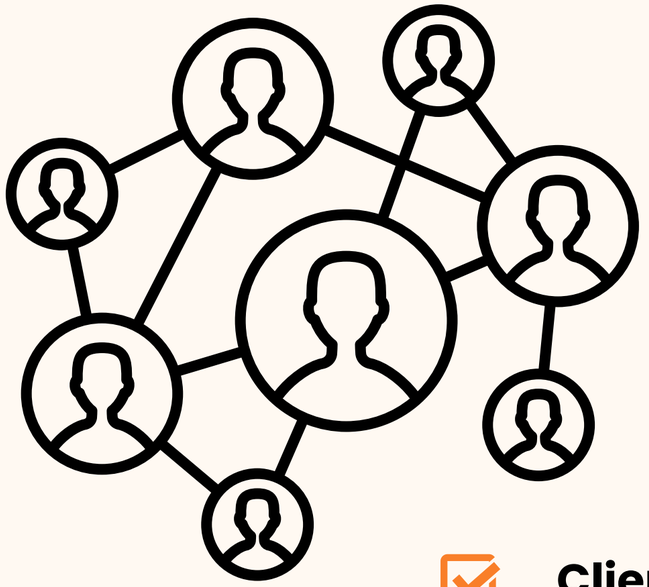
BRANCHING IN VCS

Class: BranchWorkspace
Isolation: Every branch is an object containing its own independent History Stack and Future Stack.
Behavior:

- Switching branches changes the user's active context.
- Changes in "feature-branch" do not affect "master" until a MERGE occurs.



How the clients work on file in VCS ???

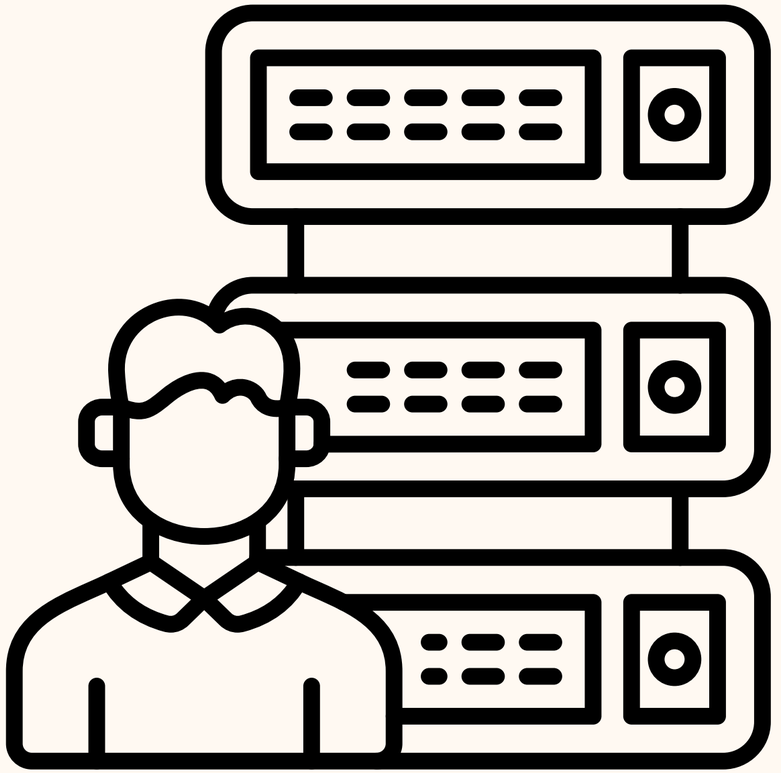


- [x] Client
- [x] Server

HOW NETWORKING IS INCLUDED IN VCS ???

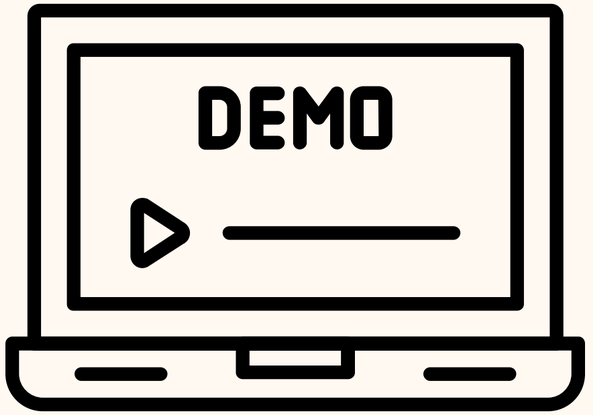
Server-Client Scripting

- How is the connection between Server & client?
- Why python socket module?
- What is Broadcasting ?? When Done ??



Broadcasting

- When a user COMMITs, the server broadcasts a notification to all other connected clients using a thread-safe loop.
- Similarly, Goes while Mergeing
- Includes error handling to remove "dead" sockets (disconnected users).



Demonstartion

Challenges

- Handling race conditions in networking.
- Managing independent history stacks for multiple branches.
- Managing the connection flow and Right time broadcasting .

Resourse

- Learn Python and OOPs From Faculty Shivangi Matedia
- Learn DSA From Faculty Madonna Lamin
- Understand client and server connection and relative networking from https://realpython.com/python-sockets/.

Thank You

Tech Titans